NPS-CS-02-002

# NAVAL POSTGRADUATE SCHOOL
## Monterey, California


**Remote Boot of a Diskless Linux Client for Operating System Integrity**

by

Bruce Allen

July 2002



Prepared for: Defense Advanced Research Project Agency
Advanced Technology Organization

This page left intentionally blank

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RADM Admiral David R. Ellison
Superintendent

R. Elster
Provost

This report was prepared by the Naval Postgraduate School Center for Information Systems Security (INFOSEC) Studies and Research (NPS CISR). Support for this work was provided by the Defense Advanced Research Projects Agency/Advanced Technology Organization.

This report was prepared by:

_____
Bruce Allen
Research Associate

Reviewed by:

_____
Neil C. Rowe
Professor
Department of Computer Science

Released by:

_____
Christopher Eagle, Chair
Department of Computer Science

_____
D. W. Netzer
Associate Provost and
Dean of Research

# REPORT DOCUMENTATION PAGE

Form approved

OMB No 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | 8 July 2002 | Technical Report |

**4. TITLE AND SUBTITLE**

Remote Boot of a Diskless Linux Client for Operating System Integrity

**5. FUNDING**

MIPR No. 00-E583

**6. AUTHOR(S)**

Bruce Allen

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Center for Information Systems Security Studies and Research (CISR)
Naval Postgraduate School, 833 Dyer Road, Monterey, CA 93943

**8. PERFORMING ORGANIZATION REPORT NUMBER**

NPS- CS- 02- 002

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

DARPA/ATO
CHATS Program
3701 North Fairfax Drive
Arlington, VA 22203-1714

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**

Approved for public release; distribution is unlimited

**12b. DISTRIBUTION CODE**

**13. ABSTRACT (Maximum 200 words.)**

These instructions describe how to build a diskless client with dedicated user storage on the server. The diskless Linux client is organized to provide read-write files over NFS at /home, read-only files over NFS for accessing bulky immutable utilities, and some volatile RAM disk files to allow the Linux Kernel to boot. This approach is exceptional because 1) the client system is diskless; system administration is managed at a single server, and 2) it is more secure; the only data that can be modified is user data under the /home directory. Although these instructions describe setup between a Linux client and a Linux server, the Linux client can boot from any server with BOOTP, TFTP and NFS services installed.

**14. SUBJECT TERMS**

Linux, remote boot, client server, system administration

**15. NUMBER OF PAGES**

12

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | Unclassified |

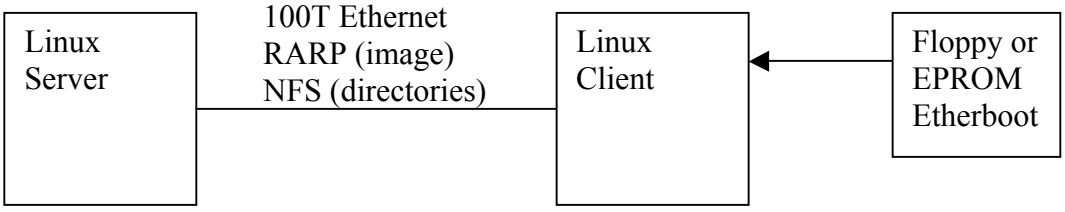
# Remote Boot of a Diskless Linux Client for Operating System Integrity

Bruce Allen

Center for Information Systems Security Studies and Research
Computer Science Department
Naval Postgraduate School
Monterey, California 93943

These instructions describe how to build a diskless client with dedicated user storage on the server. The diskless Linux client is organized to provide read-write files over NFS at /home, read-only files over NFS for accessing bulky immutable utilities, and some volatile RAM disk files to allow the Linux Kernel to boot. This approach is exceptional because 1) the client system is diskless; system administration is managed at a single server, and 2) it is more secure; the only data that can be modified is user data under the /home directory. Although these instructions describe setup between a Linux client and a Linux server, the Linux client can boot from any server with BOOTP, TFTP and NFS services installed.

The following diagram shows the basic setup. Primary components are the Linux server and the Linux client. The Linux server contains the client's Kernel and RAM Disk boot image, read-only directories, and the read-write /home directory. The server is set up with RARP services configured for providing the boot image, and with NFS services configured for providing the read-only and read-write directories. The diskless client is set up with a boot floppy or ROM to enable booting using Etherboot. Etherboot loads and unzips the Kernel image and RAM Disk, and boots the Kernel:




Linux server provides:
- Boot image: Linux Kernel + RAM Disk via RARP
- Read-only directories via NFS
- Read-write /home directory via NFS

Linux client imports:
- Etherboot from floppy
- Image from server

- Kernel from image
- RAM Disk from image

**Build the Server System**

The example shown here was developed on a X86 PC server connected to an X86 PC diskless client via Ethernet. The server was installed from the Red Hat Linux 7.1 two-CD set. Our configuration included the following selections:

- 2-button PS/2 mouse.
- Custom system installation set to install all packages from CD.
- Manually partition the disk drive using Disk Druid, defining 1) a partition of type "Linux swap", size 256M, 2) a partition of type "Linux native" at mount point "/boot" of 16M, and 3) a partition of type "Linux native" at mount point "/" using remaining space.
- Select to build without boot disk and without LILO.
- Set network to not use DHCP and to activate on boot. Set IP=192.168.0.101, netmask=255.255.255.0, network=192.168.0.0, broadcast=192.168.0.255, hostname=arrow, gateway undefined, primary DNS=192.168.0.1, secondary undefined, and tertiary undefined.

**Prepare a Boot Floppy or ROM**

1. Get latest Etherboot and doc files from Etherboot.com.
2. Uncompress these files: tar –zxvf etherboot-5.0.6.tar.gz; tar –zxvf etherboot-doc-5.0.6.tar.gz. This will create directories etherboot-5.0.6 and doc. Refer to text in doc/text for info on kgcc, the Ethernet card, and creating boot disks, if needed.
3. Edit "etherboot-5.0.6/src/Config" by adding "-ASK_BOOT=n" to the compiler options variable to prevent the boot sequence from asking how to boot.
4. A 10-second delay may be evident after the very first UDP transmit because the PC isn't initialized enough to return a successful await-reply. Etherboot assumes congestion because of this failure, and delays as per RFC2131. A successful await-reply may be achieved which eliminates this network delay by changing Etherboot code to send a second UDP transmit right after the first in src/main.c. Here is a snippet showing a second udp_transmit call in routine bootp:

```
for (retry = 0; retry < MAX_BOOTP_RETRIES; ) {
        long timeout;

        /* Clear out the Rx queue first.  It contains nothing of
         * interest, except possibly ARP requests from the DHCP/TFTP
         * server.  We use polling throughout Etherboot, so some time
         * may have passed since we last polled the receive queue,
         * which may now be filled with broadcast packets.  This will
         * cause the reply to the packets we are about to send to be
         * lost immediately.  Not very clever.  */
        await_reply(AWAIT_QDRAIN, 0, NULL, 0);
```

```
                udp_transmit(IP_BROADCAST, BOOTP_CLIENT,
BOOTP_SERVER,
                        sizeof(struct bootpip_t), &ip);
/* added second udp_transmit to get positive await_reply. */
                udp_transmit(IP_BROADCAST, BOOTP_CLIENT,
BOOTP_SERVER,
                        sizeof(struct bootpip_t), &ip);
                timeout = rfc2131_sleep_interval(TIMEOUT, retry++);
#ifdef  NO_DHCP_SUPPORT
```

5. Compile Etherboot by typing "make" from the etherboot-5.0.6 directory.
6. Etherboot 5.0.6 is not compatible with the Red Hat Linux 7.1 C compiler, so install and use the kgcc compiler instead. Install kgcc by using "rpm –i <pkg>" to install the two packages: compat-glibc-6.2-2.1.3.2.i386.rpm and compat-egcs-6.2-1.1.2.14.i386.rpm. Change file "etherboot-5.0.6/Config" to use "kgcc" instead of "cc": "cc32 = kgcc".
7. Compiling Etherboot by "make" creates many Etherboot images for supporting a variety of Ethernet cards, and for floppy, ROM and PXE ROM images. To make an Etherboot floppy for the 3c905B-TXNM board, insert a floppy and type "make bin32/3c905b-tpo100.fd0". The tpo100 is for the 3c905B-TXNM board. The fd0 makes a floppy instead of a ROM or PXE ROM burnable image.
8. Test the floppy on the target client. Since the server is not set up yet, expect "looking for BOOTP server". Look for the client MAC address. This value will be needed by file dhcpd.conf when configuring the server with DHCP services.

**Compile the Kernel**

Compile the client Kernel with switches set to compile in support for RAM Disk, BOOTP, NFS, and an Ethernet card:

1. Select the Linux base directory, /usr/src/linux-2.4.2.
2. Type "make menuconfig". This is a GUI for building configuration file include/linux/.config which specifies what is compiled into the Linux Kernel. Select block devices | RAM disk support and block devices | initial RAM disk (initrd) support. Select Networking options | IP kernel level autoconfiguration | bootp. Do not select RARP. Select NFS services via File systems | Network File Systems | NFS file system support | Provide NFSv3 client support. Select the Ethernet driver for your card; for a 3c9xx card, select Network device support | Ethernet (10 or 100Mbit) | 3COM cards | 3c590/3c900 series. Some services are not necessary and may be disabled. For example disable SCSI support via SCSI support | SCSI support. If a serial console is desired (see section "Debugging"), enable Character devices | support for console on serial port. Save this configuration. Verify the .config file by visually inspecting it.
3. Compile the Kernel: make dep, make clean, make bzImage.

**Configure Server BOOTP and TFTP Services**

Note: server name: cogswell
server IP: 192.168.0.101

client name: rosie
domain name: nps.cisrlab.com
client IP: 192.168.0.100
client MAC: 0x00105a1ed962

1. DHCP provides bootp services. Create file /etc/dhcpd.conf as shown below. This file identifies the client that the server will connect to, and what file will be downloaded. Fields domain-name-servers and domain-name identify the server. Field routers identifies the server. Field server-name and server-identifier identifies the server. Fields hardware Ethernet and fixed-address identify the client. Field filename identifies the path to the Kernel image with respect to /tftpboot. See "man dhcpd.conf" for further details on this file.

```
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.0.255;
option domain-name-servers 192.168.0.101;
option domain-name "cisrlab.com";
allow bootp;
deny unknown-clients;
subnet 192.168.0.0 netmask 255.255.255.0 {
        option routers 192.168.0.101;
}
group {
  default-lease-time -1;
  server-name "cogswell.cisrlab.com";
  server-identifier 192.168.0.101;
  use-host-decl-names on;
  host rosie {
       hardware ethernet 00:10:5A:1E:D9:62;
       fixed-address 192.168.0.100;
       filename "/rosie/zImage";
  }
}
```

2. DHCP requires that file dhcpd.leases exists. Create this file by typing "touch /var/lib/dhcp/dhcpd.leases".
3. Enable dhcpd by typing "linuxconf" and selecting: control | control panel | control service activity | dhcpd | startup | automatic, and then selecting accept. Close linuxconf.
4. Enable TFTP services by editing /etc/xinetd.d/tftp and changing line "disable = yes" to "disable = no".
5. Edit /etc/hosts.allow to enable TFTP services by adding tftp, portmap and statd content. The file should read as follows:

```
#
# hosts.allow   This file describes the names of the hosts which are
#               allowed to use the local INET services, as decided
#               by the '/usr/sbin/tcpd' server.
#
tftp:ALL
portmap: .cisrlab.com
statd: .cisrlab.com
```

6. Create directory /tftpboot/rosie. The DHCP server provides the Kernel boot image from this path.
7. Create the following /etc/hosts file:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain localhost
192.168.0.101 cogswell.cisrlab.com  cogswell
192.168.0.100 rosie.cisrlab.com       rosie
192.168.0.1     netframe5.cisrlab.comnetframe5
```

8. Turn off the Linux firewall by putting the following lines at the bottom of initialization script /etc/rc.d/rc.local:

```
ipchains -F
ipchains -P input ACCEPT
ipchains -P output ACCEPT
ipchains -P forward ACCEPT
```

**Configure NFS Services**

The server must be configured to export NFS files and the client must be configured to mount NFS files. The following directories are made available via NFS:

**/bin, /lib, /sbin**

These read-only directories provide all the utilities and libraries from the /bin, /lib, and /sbin directories of the server. When these directories are mounted, the equivalent bootstrapping directories on the RAM Disk become inaccessible.

**/home**

This read-write directory provides persistent storage of files on the server for the user.

**/usr**

This read-only directory contains general utility files available for run-time applications.

1. Enable NFS by typing "linuxconf" and selecting: control | control panel | control service activity | nfs | startup | automatic, and then selecting accept. Close linuxconf.
2. Create file /etc/exports so that the server can export required files. The /etc/exports file should look like this:

```
/bin          rosie.cisrlab.com(ro,insecure)
/home         rosie.cisrlab.com(rw,no_root_squash,insecure)
/lib          rosie.cisrlab.com(ro,insecure)
/sbin         rosie.cisrlab.com(ro,insecure)
/usr          rosie.cisrlab.com(ro,insecure)
```

**Enable Server BOOTP, TFTP and NFS Services**

1. Reboot to make the BOOTP, TFTP and NFS services active.
2. Verivy that ipchains are configured for "accept": ipchains -L
3. Verify that the BOOTP, TFTP and NFS processes are active by typing "netstat -a". Observe that bootps, tftp and nfs processes are listed.
4. Verify the Ethernet configuration by typing "ifconfig –a".

**Creating Users for the Client**

1. Because the client is diskless, users cannot be created on the client. Create the client users on the server now. These users will be installed on the RAM Disk when the RAM Disk is created. Users may be updated later on the RAM Disk by copying the /etc/passwd and /etc/shadow files to the RAM Disk. Change the root user's home directory to /home/root when copying /etc/passwd to the RAM Disk. See section "Configure the RAM Disk" for instructions on changing the home directory for the root user.

**Create the RAM Disk**

1. Create a 18000K file sink called "/root/initrd": dd if=/dev/zero of=/root/initrd bs=18000k count=1
2. Create the actual "/root/initrd" file: mke2fs -N 20000 -F -m0 /root/initrd.
3. Create the initrd mount path: mkdir /mnt_rd. Now /mnt_rd may be used to freely mount and unmount file /root/initrd as a RAM Disk loop device.
4. Mount the RAM disk with the following script:

```
#! /bin/sh
mount -t ext2 -o loop /root/initrd /mnt_rd
```

5. Unmount the RAM disk using the following script. Note that this script contains the e2fsck command which repairs the image if it is broken.

```
#! /bin/sh
sync; sync
e2fsck /dev/loop0
```

umount /mnt_rd

**Populate the RAM Disk**

The RAM Disk must have enough files and directories for Linux to boot and mount complete file system resources via NFS. The RAM Disk contains the following empty, partially filled, and completely installed directories:

| **empty** | **partial** | **complete** |
|---|---|---|
| /home | /bin | /dev |
| /proc | /lib | /etc |
| /tmp | /sbin | /mnt |
| /usr | | |
| /var | | |

These directories are as follows:

**/home**

This directory must exist during bootup, and becomes inaccessible when the server's /home is mounted via NFS. The server's /home is mounted read-write and provides space for persistent user data.

**/proc, /tmp, /usr, and /var**

These empty directories must exist for Linux to boot. The /var directory contains many empty subdirectories, as required by various Linux packages.

**/bin, /lib, and /sbin**

These large directories are filled with enough files to allow NFS to work. Once Linux has booted enough to use NFS, the server's /bin, /lib, and /sbin directories are mounted, and these "bootstrapping directories" become inaccessible.

**/dev**

For simplicity, all devices are copied onto the RAM Disk. Most of the devices are not used and may be deleted, but deleting devices that are required can produce failures that are difficult to track down. If desired, devfs may be used. devfs dynamically creates potentially useful devices during bootup. If devfs is used, all devices may be deleted (but /dev must remain). Use of devfs is discouraged because devfs increases the size of the Kernel and takes longer to initialize. devfs is enabled by compiling the Kernel with the following switches turned on in file .config: CONFIG_EXPERIMENTAL=y, CONFIG_DEVFS_FS=y, CONFIG_DEVFS_MOUNT=y, and CONFIG_DEVFS_DEBUG=y. If these switches are set and devfs is not desired, devfs may be disabled by passing "devfs=nomount" at the kernel boot command line. See sections "Compile the Kernel" and "Prepare the Downloadable ZIP image" for the location of the .config file and for passing boot parameters to the Kernel.

**/etc**

For simplicity, all these files are copied to the RAM Disk. Many of these files are unnecessary, but since /etc is used during run-time, it is difficult to determine which files are unnecessary. I chose to use this /etc during run-time rather than mount the server's /etc via NFS because X-Windows comes up faster when it can reference more files locally.

**/mnt**
This tiny directory defines mount paths. It can be modified since it is on the volatile RAM Disk. The server's /mnt directory could be used via NFS, and even mounted RW rather than RO, but it is cleaner to have only one RW directory on the server at /home. My current idea is to have hardcoded mount paths under /mnt on the RAM Disk, and persistent mount paths under /home.

The RAM Disk also contains empty file /mnt_rd/fastboot. By being present, the client does not take time to check the integrity of the RAM Disk while booting. It is not necessary for the client to check the integrity of the RAM disk because the client cannot corrupt it.

Install directories and files on the RAM Disk as follows.

1. Mount the RAM Disk using the script defined in section "Create the RAM Disk".
2. Install directories and files by executing the following script:

```
#! /bin/sh
# create the fastboot file to avoid unnecessarily checking the integrity
# of the RAM Disk.  See "man fastboot".
touch /mnt_rd/fastboot
chattr -i /mnt_rd/fastboot

# create directories
for i in bin home lib lib/i686 proc sbin tmp usr var var/arpwatch var/cache
var/db var/gdm var/lib var/local var/lock var/lock/subsys var/log
var/log/news var/nis var/opt var/preserve var/run var/run/netreport
var/spool var/spool/anacron var/spool/mqueue var/state var/tmp var/yp
do
mkdir /mnt_rd/$i
done
#
# put minimal files into bin, lib and sbin
for i in bin/basename bin/bash bin/cat bin/chgrp bin/chmod bin/date
bin/dmesg bin/echo bin/egrep bin/false bin/gawk bin/grep bin/hostname
bin/loadkeys bin/ls bin/more bin/mount bin/netstat bin/ping bin/ps
bin/pwd bin/rm bin/sed bin/sleep bin/touch bin/true bin/umount bin/uname
lib/i686/libc-2.2.2.so lib/i686/libm-2.2.2.so lib/ld-2.2.2.so lib/libcrypt-
2.2.2.so lib/libdl-2.2.2.so lib/libnsl-2.2.2.so lib/libnss_files-2.2.2.so
lib/libnss_nisplus-2.2.2.so lib/libproc.so.2.0.7 lib/libresolv-2.2.2.so
lib/libtermcap.so.2.0.8 sbin/consoletype sbin/devfsd sbin/getkey
```

```
sbin/hwclock sbin/ifconfig sbin/ifdown sbin/ifup sbin/init sbin/initlog
sbin/insmod sbin/killall5 sbin/klogd sbin/minilogd sbin/mkkerneldoth
sbin/portmap sbin/route sbin/rpc.lockd sbin/rpc.statd sbin/runlevel
sbin/service sbin/setsysfont sbin/swapon sbin/sysctl sbin/syslogd
do
cp -a /$i /mnt_rd/$i
done

# put some files from /usr into bin also
cp -a /usr/bin/env /mnt_rd/bin/env
cp -a /usr/bin/tail /mnt_rd/bin/tail
cp -a /usr/sbin/rpcinfo /mnt_rd/bin/rpcinfo

# reduce the size of lib files
strip /mnt_rd/lib/*.*
strip /mnt_rd/lib/i686/*.*

# make symbolic links for some files in bin, lib and sbin
cd /mnt_rd/bin
ln -s gawk awk
ln -s bash sh
cd /mnt_rd/lib/i686
ln -s libc-2.2.2.so libc.so.6
ln -s libm-2.2.2.so libm.so.6
cd /mnt_rd/lib
ln -s ld-2.2.2.so ld-linux.so.2
ln -s libcrypt-2.2.2.so libcrypt.so.1
ln -s libdl-2.2.2.so libdl.so.2
ln -s libnsl-2.2.2.so libnsl.so.1
ln -s libnss_files-2.2.2.so libnss_files.so.2
ln -s libnss_nisplus-2.2.2.so libnss_nisplus.so.2
ln -s libresolv-2.2.2.so libresolv.so.2
ln -s libtermcap.so.2.0.8 libtermcap.so.2
cd /mnt_rd/sbin
ln -s insmod modprobe
ln -s killall5 pidof
ln -s init telinit

# fully populate directories for dev, etc and mnt
cp -a /dev /mnt_rd/dev
cp -a /etc /mnt_rd/etc
cp -a /mnt /mnt_rd/mnt
```

**Configure the RAM Disk**

At this point, the bulk of the RAM Disk is ready, but some files will need to be changed in /mnt_rd/etc to make the client behave like a client rather than the server that these files were copied from.

1. Configure file /mnt_rd/etc/fstab to mount required directories via NFS so that it reads as follows:

| | | | | | |
|---|---|---|---|---|---|
| cogswell.cisrlab.com:/bin | /bin | nfs | ro | 0 | 0 |
| cogswell.cisrlab.com:/home | /home | nfs | defaults | 0 | 0 |
| cogswell.cisrlab.com:/lib | /lib | nfs | ro | 0 | 0 |
| cogswell.cisrlab.com:/sbin | /sbin | nfs | ro | 0 | 0 |
| cogswell.cisrlab.com:/usr | /usr | nfs | ro | 0 | 0 |
| /dev/fd0 | /mnt/floppy | auto | noauto,owner | 0 | 0 |
| none | /proc | proc | defaults | 0 | 0 |
| none | /dev/pts | devpts | gid=5, mode=620 | 0 | 0 |
| /dev/cdrom | /mnt/cdrom | iso9660 | noauto, owner, kudzu, ro | 0 | 0 |
| /dev/hdd4 | /mnt/zip100.0 | auto | noauto, owner, kudzu | 0 | 0 |
| LABEL=/ | / | ext2 | defaults | 1 | 1 |

2. Remove files /mnt_rd/etc/dhcpd.conf, /mnt_rd/etc/exports, /mnt_rd/etc/mtab and /mnt_rd/etc/resolv.conf.
3. Edit file /mnt_rd/etc/hosts and remove the line “192.168.0.1 netframe5.cisrlab.comnetframe5”.
4. Edit file /mnt_rd/etc/hosts.allow and remove lines “portmap: .cisrlab.com” and “statd: .cisrlab.com”.
5. Edit /mnt_rd/etc/passwd and change the home directory of the root user to be under /home/root rather than /root. Do this by changing the root entry to read “root:x:0:0:root:/home/root:/bin/bash”. This change is made because the client is restricted to read/write access under /home.
6. Change /mnt_rd/etc/sysconfig/network to read as follows:

```
NETWORKING=yes
HOSTNAME=rosie
GATEWAY=192.168.0.101
```

7. Change IPADDR in /mnt_rd/etc/sysconfig/network-scripts/ifcfg-eth0 to read “IPADDR=192.168.0.100”.
8. Disable X-Windows, if desired, by changing file etc/inittab line “id:5:initdefault” to read “id:3:initdefault”.
9. The RAM Disk image is now ready to use. See section “Configure X-Windows” for configuring X-Windows on the client.

**Prepare the Downloadable ZIP Image**

The image that Etherboot downloads must be a tagged image. Utility mknbi-linux prepares this image. Get and install mknbi-linux as follows:

1. Get mknbi-1.2-7.noarch.rpm from sourceforge.net/projects/etherboot.
2. Install mknbi-1.2-7.noarch.rpm by typing "rpm -i mknbi-1.2-7.noarch.rpm".

The tagged image that mknbi-linux crates contains the zipped Linux Kernel image, the zipped RAM disk image, and specific mknbi and Linux command parameters.

1. Run the following script to create tagged image file /tftpboot/rosie/zImage. Note the commented out line which can be used to enable the console on both the monitor and the serial port.

```
#! /bin/sh
gzip -c initrd > initrd_gz
mknbi-linux -ip=rom -rootdir=/dev/ram0 --append="ramdisk=18000k
nousb devfs=nomount" -o /tftpboot/rosie/zImage /usr/src/linux-
2.4.2/arch/i386/boot/bzImage initrd_gz
# mknbi-linux -ip=rom -rootdir=/dev/ram0 --append="console=tty
console=ttyS0,9600 ramdisk=18000k nousb devfs=nomount" -o
/tftpboot/rosie/zImage /usr/src/linux-2.4.2/arch/i386/boot/bzImage
initrd_gz
```

2. Boot the client. See section "Debugging" for some debugging approaches.

**Configure X-Windows**

1. Configure X-Windows on the client and save the configuration to default path /etc/X11/XF86Config. Note that this path is on the client's volatile RAM Disk.
2. Run "startx" to ensure that X-windows is working.
3. Copy the newly created XF86Config file to the server's permanent RAM Disk image at /mnt_rd/etc/X11/XF86Config so that X-Windows will use this file when it starts. Do this by copying /etc/X11/XF86Config on the client to /home/root/XF86Config, which is shared. Then, on the server, mount the RAM Disk, copy /home/root/XF86Config to /mnt_rd/etc/X11/XF86Config, and then unmount the RAM Disk.

**Debugging**

- Watch what is displayed on the console as Linux boots. Early text cannot be directed to a file, but it can be directed through a serial port to a terminal emulator with a big screen buffer. The following steps describe setup for monitoring output using teraterm:
  1. Turn on serial devices and serial console support using "make menuconfig", and recompile the Kernel. See "Compile the Kernel".
  2. Update the script which prepares the downloadable image so that the Kernel boot parameters include the text "console=ttyS0,9600". The commented out example

in section “prepare the downloadable ZIP image” shows the console and serial port both active at the same time.

3. Connect a PC to PC serial cable between the client and the third “terminal emulator” PC. A null modem will be required if the cable is straight (i.e. intended for a printer).
4. Install teraterm on the terminal emulator PC.
5. Open teraterm. Configure it to use the serial port and specify the baud rate. Make the window size larger and increase the buffer size to 800 lines. Save the configuration by name so that it can be restored later. When needed, clear the window pane by clearing the buffer.

- Examine Ethernet traffic. Use a crossover cable or a dedicated hub to eliminate other LAN traffic, and try “tcpdump -vnN”.
- Commands and configuration files have man pages. Reference them and their “see also” pages.
- Utilities such as etherboot and mknbi have detailed README files. Reference them and any documents they cite.
- Check mailing lists for related keywords and error strings that might show up.

**Known errors**

Modules fstab, atd, lpd and Linuxconf hooks do not load correctly. These errors are likely a result of missing directories or wrong owners under /var.

**INITIAL DISTRIBUTION LIST**

| | | |
|---|---|---|
| 1. | Defense Technical Information Center<br>8725 John J. Kingman Rd., STE 0944<br>Ft. Belvoir, VA 22060-6218 | 2 |
| 2. | Dudley Knox Library, Code 013<br>Naval Postgraduate School<br>Monterey, CA 93943-5100 | 2 |
| 3. | Research Office, Code 09<br>Naval Postgraduate School<br>Monterey, CA 93943-5138 | 1 |
| 4. | Dr. Douglas Maughan<br>Defense Advanced Research Projects Agency<br>Advanced Technology Office<br>3701 North Fairfax Drive<br>Arlington, VA 22203-1714 | 1 |
| 5. | Dr. Cynthia E. Irvine<br>Code CS/Ic<br>Department of Computer Science<br>Naval Postgraduate School<br>Monterey, CA 93943-5118 | 10 |
| 6. | Bruce Allen<br>Department of Computer Science<br>Naval Postgraduate School<br>Monterey, CA 93943-5118 | 2 |